uCAN Programming

English Manual

2014.07.17

Version 1.0

Serial Communication Experts
SystemBase
Since 1987

History

Date (MM/DD/YYYY)	Version	Revised	Description
07/16/2014	1.0	All	New

Website: http://www.sysbas.com/

Technical Support: http://www.solvline.com/

Phone: 82-2-855-0501 FAX: 82-2-855-0580

16F Daerung Post Tower-1, 288, Digital-ro, Guro-gu, Seoul, Republic of Korea

If you have any questions, please post your inquiries at Solvline. (http://www.solvline.com/)

Index

Index

1. Overview

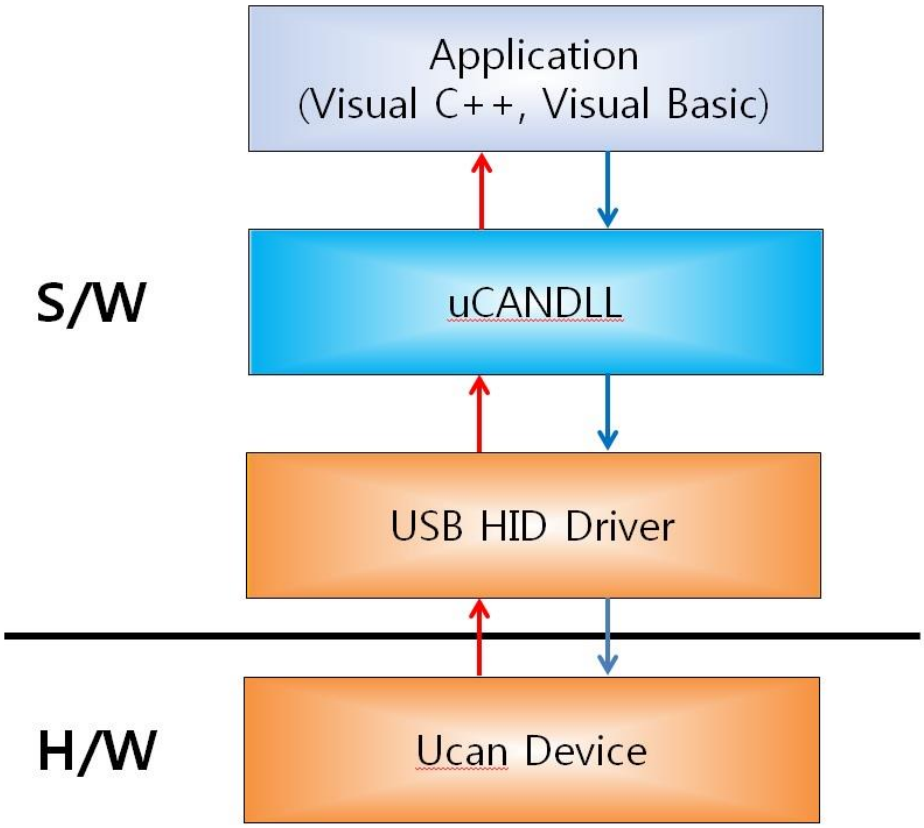
● DLL

Dynamic-link library or DLL is a shared library for Microsoft Windows. The DLL has functions and data types that can call other programs. Most DLL file extensions are named “.dll”. When DLL files are used, the development speed can be achieved faster with better reliability.

● uCAN DLL

uCAN DLL provides DLL for those who use Visual C++ or Visual Basic to directly access uCAN device. Transmitting/Receiving CAN frame, and setting configuration can be achieved.

● uCANDLL Structure



● uCANDLL File and Folder Structure

File and Directory	Content
VC Sample\	Visual C++ Sample Code
uCANDLL.dll	DLL File
uCANDLL.lib	Library File
uCANDLL.h	Header File
uCANDLL.pdf	DLL Manual File

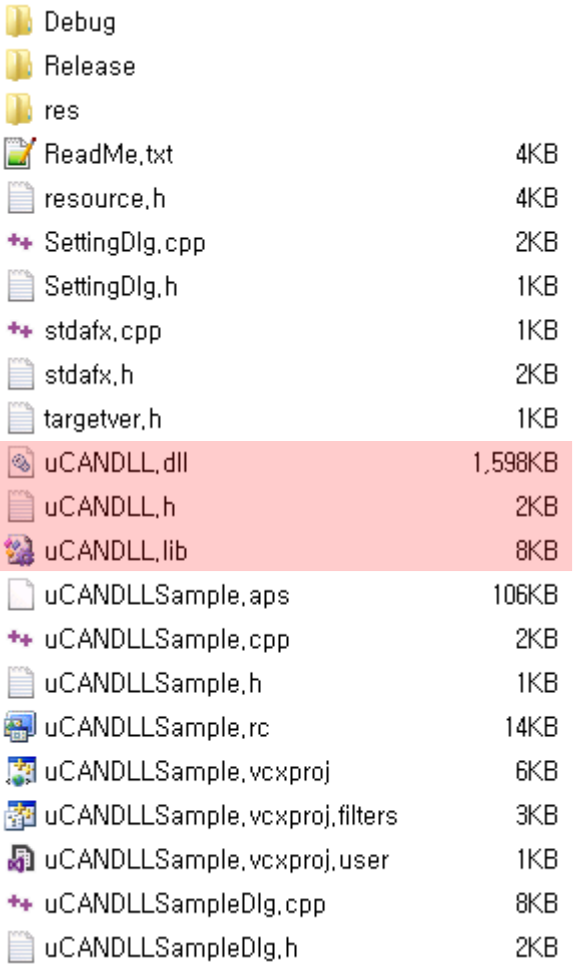
● IMPORTANT NOTE

When receiving CAN frame, event message is transmitted. To initialize uCANDLL, handle for receiving CAN frame receive event message are passed by the handle of the argument value.

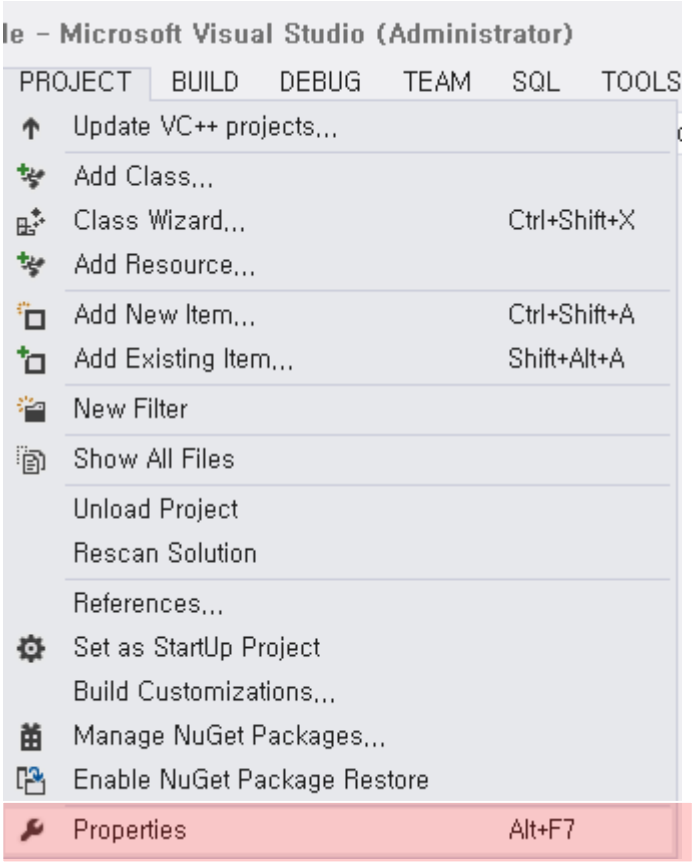
2. Usage

● Visual C++

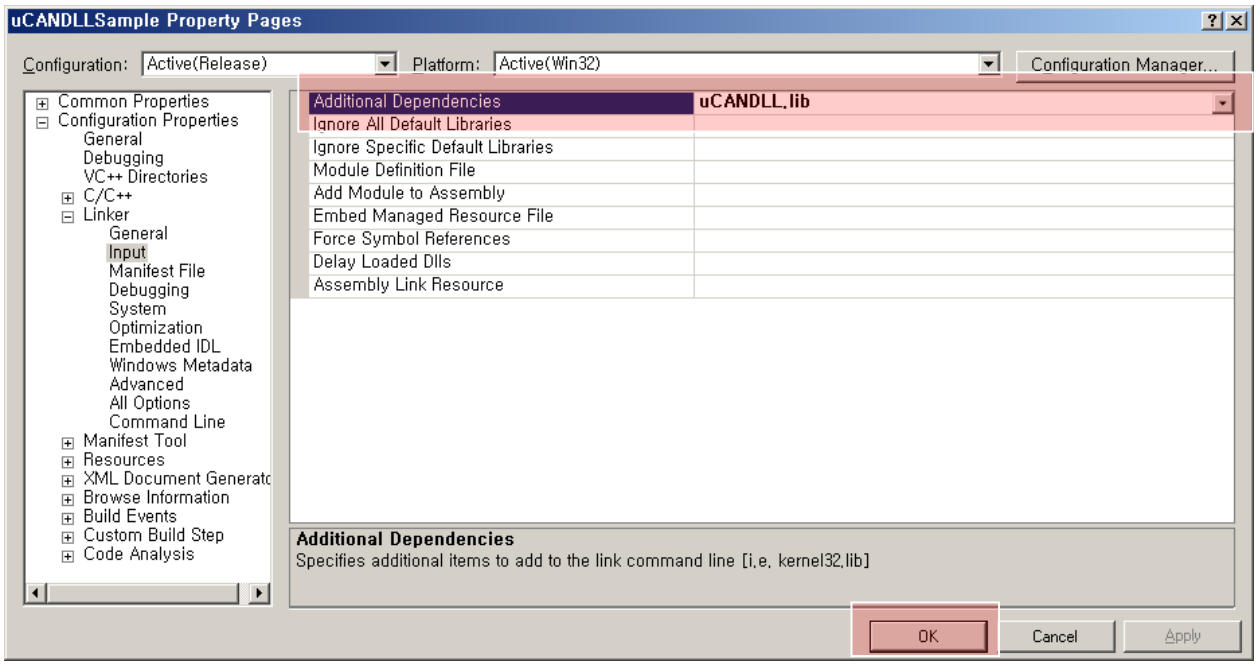
1. Move uCANDLL.dll, uCANDLL.lib, and uCANDLL.h files to source code folder as shown below.



2. From the menu in the Visual Studio, click “PROJECT” then “Properties” to open Property window.



3. From the left side, click “Linker” then “Input”. Type “uCANDLL.lib” next to “Additional Dependencies” and click “OK” button.



4. Include uCANDLL.h header file.

```
#include "stdafx.h"
#include "uCANDLLSample.h"
#include "uCANDLLSampleDlg.h"
#include "afxdialogex.h"
#include "SettingDlg.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#endif


#include"uCANDLL.h"
```

5. Add event function to message map.

uCANDLLSampleDlg.h

```
CuCANDLLSampleDlg
    ULONGLONG m_TxCountValue;
    ULONGLONG m_RxCountValue;

protected:
    afx_msg LRESULT OnCANRx(WPARAM wParam, LPARAM lParam);
    afx_msg LRESULT OnCANError(WPARAM wParam, LPARAM lParam);
    afx_msg LRESULT OnCANTx(WPARAM wParam, LPARAM lParam);
};
```

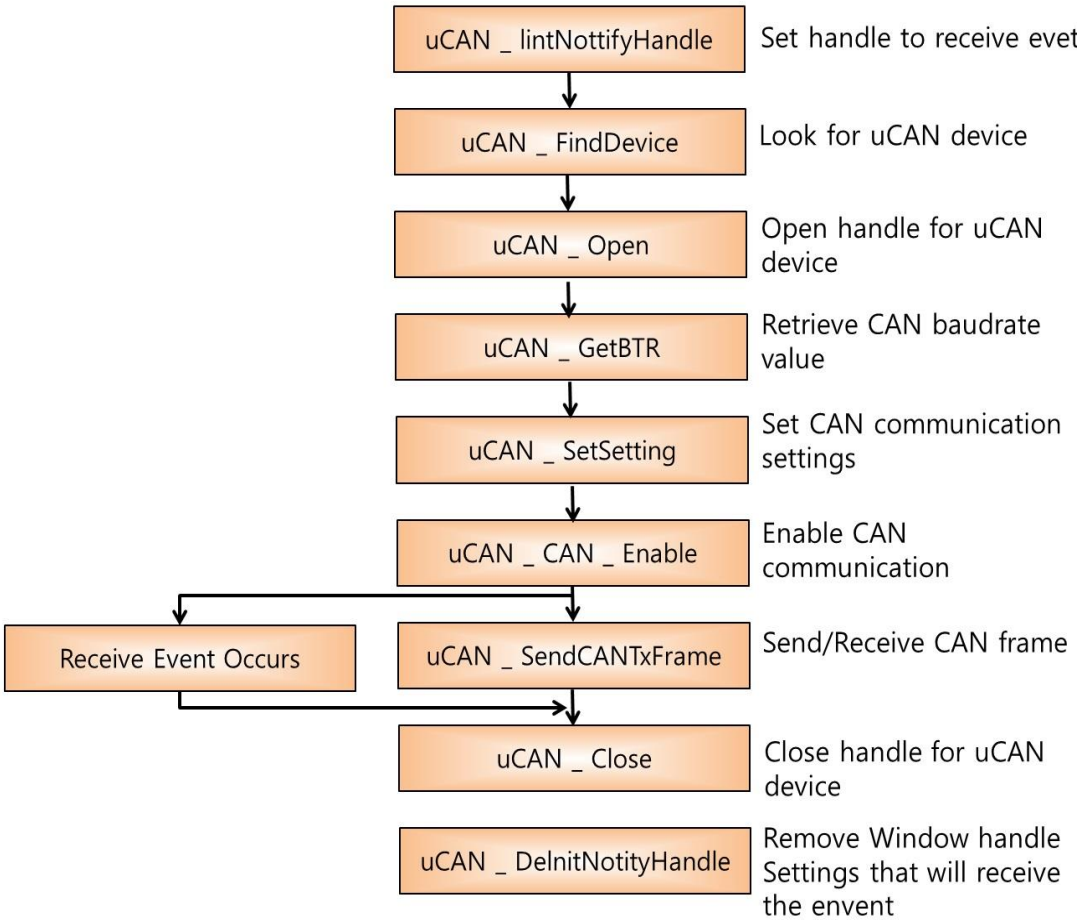
uCANDLLSampleDlg.cpp | uCANDLLSampleDlg.h

```
CuCANDLLSampleDlg


BEGIN_MESSAGE_MAP(CuCANDLLSampleDlg, CDialogEx)
    ON_MESSAGE(ON_CANRX, &CuCANDLLSampleDlg::OnCANRx)
    ON_MESSAGE(ON_CANERROR, &CuCANDLLSampleDlg::OnCANError)
    ON_MESSAGE(ON_CANTX, &CuCANDLLSampleDlg::OnCANTx)
END_MESSAGE_MAP()
```

6. Start coding according to API as shown below.

```
int num;
CString tmp;

num = uCAN_FindDevice();
tmp.Format(_T("CALL FUNC >> uCAN_FindDevice : %d"),num);
```

3. Program Coding Order



4. User Defined Message

● ON_CANRX

Message Parameter	Description
wParam	Received CAN_Frame structure pointer is passed
lParam	Flag is passed. If there are unprocessed CAN frame in the DLL buffer, BUSY flag is set. 0: NORMAL 2: BUSY

● ON_CANTX

Message Parameter	Description
wParam	Transmitted CAN_Frame structure pointer is passed
lParam	Flag is passed. If there are unprocessed CAN frame in the DLL buffer, BUSY flag is set. 0: NORMAL 2: BUSY

● ON_CANERROR

Message Parameter	Description
wParam	CAN_Error structure pointer is passed
lParam	None

5. Structure Data Type

● CAN_Frame

Structure Member Variable	Description
UINT8 Format	Shows CAN Frame types and Transmit/Receive 0x04: (Receive)Standard Data 0x05: (Receive)Standard Remote 0x06: (Receive)Extended Data 0x07: (Receive)Extended Remote 0x14: (Transmit)Standard Data 0x15: (Transmit)Standard Remote 0x16: (Transmit)Extended Data 0x17: (Transmit)Extended Remote
UINT32 ID	CAN Frame ID (Hex)
UINT8 DLC	CAN Frame DLC (0 ~ 8)
UINT8 Data[8]	CAN Frame Data Array (8 bytes)
UINT32 TimeStamp	CAN Frame Time Stamp (Receive only)

● CAN_BTR

Structure Member Variable	Description
UINT8 BTR	BTR in CAN baud rate
UINT8 SJW	SJW in CAN baud rate
UINT8 TSEG1	TSEG1 in CAN baud rate
UINT8 TSEG2	TSEG2 in CAN baud rate
UINT8 CLKDIV	CLKDIV in CAN baud rate

● CAN_Mask

Structure Member Variable	Description
UINT32 ID	ID to mask Receive ID
UINT32 MASK	MASK to mask Receive ID

Structure Data Type

UINT8 Format	Format to mask Receive ID 0: ALL 1: Standard 2: Extended

● CAN_Error

Structure Member Variable	Description
UINT8 TEC	TEC(Transmit Error Count) in uCAN
UINT8 REC	REC(Receive Error Count) in uCAN
UINT8 LEC	LEC(Last Error Count) in uCAN 0x00: No error 0x01: Stuff error 0x02: Form error 0x03: Ack error 0x04: Bit error 0x05: CRC error
UINT8 Mode	Error status in uCAN 0x00: active mode 0x01: warning mode 0x02: error passive mode 0x03: bus-off mode
UINT16 Stuff_EC	Stuff Error Count in uCAN
UINT16 Form_EC	Form Error Count in uCAN
UINT16 Ack_EC	ACK Error Count in uCAN
UINT16 Bit_EC	Bit Error Count in uCAN
UINT16 CRC_EC	CRC Error Count in uCAN

6.API

● uCAN_SetNotifyHandle

Function Prototype	UINT8 uCAN_SetNotifyHandle(HANDLE hwnd)
Function	Set window handle that will take CAN frame transmit/receive event message
Argument	hwnd : handle to take transmit/receive event (in)
Return	0x00: Succeed to set window handle 0x01: Failed to set window handle

● uCAN_FindDevice

Function Prototype	UINT8 uCAN_FindDevice(int *MaxCount)
Function	Get recognized numbers(count) of uCAN device
Argument	MaxIndex: recognized numbers of uCAN device (out)
Return	0x00: Recognized the device 0x01: Cannot find the device

● uCAN_Open

Function Prototype	UINT8 uCAN_Open(int index)
Function	Open uCAN device handle
Argument	index: uCAN device index trying to open the handle (in)
Return	0x00: Connected 0x01: Index that cannot find the device 0x10: Connected to other process

● uCAN_Close

Function Prototype	UINT8 uCAN_Close (void)
Function	Close uCAN device handle

Argument	None
Return	0x00: Properly disconnected 0x01: Close Handle Error

● uCAN_SetSetting

Function Prototype	UINT8 uCAN_SetSetting(CAN_BTR btr, CAN_Mask mask, UINT8 function)
Function	Set communication settings for uCAN device
Argument	btr: CAN_BTR structure (in) mask: CAN_Mask structure (in) function: Contains information regarding other functions (in) 0x01: AR(automatic retransmission) function enable 0x02: ABOR(automatic Bus-Off recovery) function enable 0x10: Terminating resistor enable
Return	0x00: Succeed to set the configuration 0x10: Cannot connect to the device 0x20: Error while communicating with uCAN

● uCAN_GetSetting

Function Prototype	UINT8 uCAN_GetSetting(CAN_BTR *btr, CAN_MASK *mask, UINT8 *function)
Function	Get communication settings from uCAN device
Argument	btr: CAN_BTR structure (out) mask: CAN_Mask structure (out) function: Variable to get information for other functions (out) 0x01: AR(automatic retransmission) function enable 0x02: ABOR(automatic Bus-Off recovery) function enable 0x10: Terminating resistor enable
Return	0x00: Successfully retrieved 0x01: Handle is not opened 0x02: Cannot read the configuration value

● uCAN_GetBTR

Function Prototype	UINT8 uCAN_GetBTR (UINT32 baud, CAN_BTR *btr)
Function	To set CAN baud rate, CAN_BTR structure value is calculated to be retrieved.
Argument	baud: Input CAN baud-rate to be configured (in) btr: Get CAN_BTR structure values of CAN baud rate to be configured. (out)
Return	0x00: Successfully calculated 0x01: Cannot calculate the CAN_BTR value
Note	Opposite of uCAN_GetBaudrate

● uCAN_GetBaudrate

Function Prototype	UINT8 uCAN_GetBaudrate (UINT32 *baud, CAN_BTR btr)
Function	Get CAN baud-rate value with CAN_BTR value.
Argument	baud: Variable to store CAN baud-rate value. (out) btr: Get CAN_BTR structure value which is trying to get CAN baud rate value. (in)
Return	0x00: Successfully transmitted 0x01: Calculation Error
Note	Opposite of uCAN_GetBTR

● uCAN_GetSamplingPoint

Function Prototype	UINT8 uCAN_GetSamplingPoint(float *SamplingPoint, CAN_BTR btr)
Function	CAN_BTR 값으로 Get CAN SamplingPoint with CAN_BTR value.
Argument	SamlingPoint: Variable to save CAN SamplingPoint (out) btr: Get CAN_BTR structure value which is trying to get CAN SamplingPoint. (in)
Return	0x00: Successfully transmitted 0x01: Calculation Error

● uCAN_CAN_Enable

Function Prototype	UINT8 uCAN_CAN_Enable (UINT8 enable)
Function	Enable/Disable CAN communication.
Argument	enable: CAN SamplingPoint (in) 0: Disable 1: Enable
Return	0x00: Successfully transmitted 0x01: Unpermitted value 0x10: Handle value that is not opened

● uCAN_SetSerial

Function Prototype	UINT8 uCAN_SetSerial (char *strBuffer, UINT8 strBufferSize)
Function	Save uCAN Serial information.
Argument	istrBuffer: Variable that has serial information (in) strBufferSize: Size of variable which has serial information (in) (Same or less than 10)
Return	0x00: Successfully transmitted 0x01: Unpermitted value 0x10: Handle value that is not opened
Note	Opposite of uCAN_GetSerial Call uCAN_Open before using.

● uCAN_GetSerial

Function Prototype	UINT8 uCAN_GetSerial (int index, char *strBuffer, UINT8 strBufferSize)
Function	Read uCAN serial information.
Argument	index: Index value that will get serial information (in) strBuffer: Variable to store serial information (out) strBufferSize: Size of variable which will save serial information (in) (Same or greater than 10)

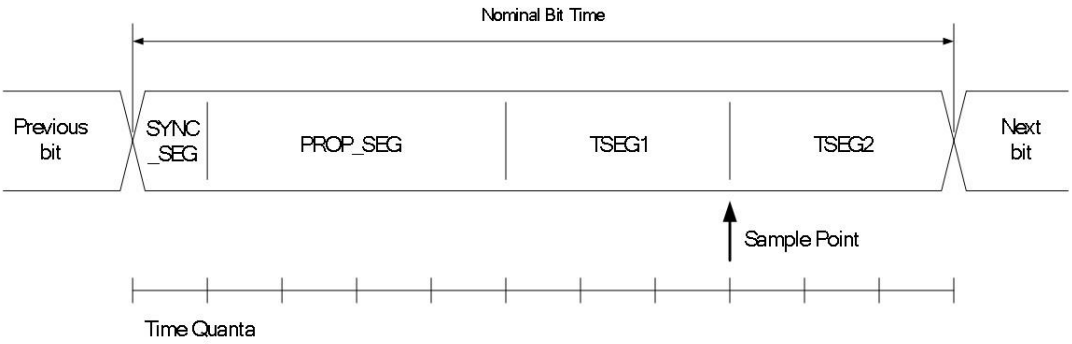
Return	0x00: Successfully transmitted 0x01: Unpermitted value 0x10: Handle value that is not opened
Note	Opposite of uCAN_SetSerial Call uCAN_FindDevice before using

● uCAN_SendCANTxFrame

Function Prototype	UINT8 uCAN_SendCANTxFrame (CANTxFrame Tx)
Function	Transmit CAN Frame
Argument	Tx: CANTxFrame structure (in)
Return	0x00: Successfully transmitted 0x01: Unpermitted Tx.Format value 0x02: Unpermitted Tx.ID value 0x04: Unpermitted Tx.DLC value 0x10: Handle value that is not opened

7. Note

● Baud-rate



Parameter	Range	Function
BRP	1~32	Set size of $1T_q$
SYNC_SEG	$1T_q$	Used in CAN Bus to synchronize various nodes. Size is fixed to $1T_q$.
PROP_SEG	(1~8) x T_q	Compensates for physical delay times. (Physical bus and internal CAN node transmission delay)
TSEG1	(1~8) x T_q	Used to correct Phase Edge Error. May be lengthened temporarily by synchronization.
TSEG2	(2~8) x T_q	Used to correct Phase Edge Error. May be shorten temporarily by synchronization.
SJW	(1~4) x T_q	Set T_q that TSEG1 may be lengthened or TSEG2 may be shortened. Should not be greater than TSEG1.

PROP_SEG + TSEG1 + 1 >= TSEG2

Calculation

CAN_CLK = 60000000 / (DIV+1)

CAN Baud rate = CAN_CLK / ((BRP+1) X (TSEG1+2+TSEG2+1))

Sample Point = (TSEG1+2) / (TSEG1+2+TSEG2+1) X 100

Setting 60~70% for Sample Point is recommended

(However, for CANOpen, set it to 80%~90%)

Example)

Baud rate	Sample Point	ClkDiv	BTR	SJW	TSEG1	TSEG2
50		3	24	3	6	7
100		2	19	3	6	6
125		1	29	3	6	7
250		1	14	3	6	7
500		1	7	3	6	6
1000		1	3	3	6	6

● Mask

Generally, in CAN communication, receive ID and receive mask ID are used to filter other CAN frames to receive desired frame only to lighten the traffic.

Receive ID indicates CAN frame ID that the user wished to see. Receive mask ID checks all the received data. If it matches the receive ID, data is received, else it is not received.